Association for Information Systems
AIS Electronic Library (AISeL)

PACIS 2007 Proceedings

Pacific Asia Conference on Information Systems (PACIS)

2007

Supporting Decision Makers in Better Dealing with Interrelaetaed Decissions

Angela Liew
University of Auckland

David Sundaram
University of Auckland

57. Supporting Decision Makers in Better Dealing with Interrelaetaed Decissions

Angela Liew
Department of Accounting and Finance, University of Auckland, New Zealand

David Sundaram
Department of Information Systems and Operations Management, University of Auckland, New Zealand

Abstract
Every decision has a different level of influence or impact in the human life. Very often, numerous smaller decisions have to be made before a complex decision can eventually arrive at its best conclusion. Moreover, each decision may have a bearing on other subsequent decisions, and thus requires the decision making process to be structured in such a flexible manner that enables the decisions to be considered and solved differently each time. However, most decision making processes and systems are designed to solve simple and linear problems and are therefore unable to support complex problems which consist of interrelated decisions that span across multiple domains, paradigms, and/or perspectives. Furthermore, the true purpose of decision making is to gain a better understanding of the issues involved behind each decision. To address these problems we first proposed conceptual decision-making and modelling processes, and then developed and implemented a flexible object-oriented decision system framework, architecture, and prototype to support these proposed processes. Through the implementation, we were able to explore and implement some general modelling ideas as well as specific issues such as the integration of models and scenarios of different types, levels of complexity, depths of integrations, and decision maker orientations.

Keywords: Decisions, Decision Making, Decision Support, Models, Model Management

Decisions and Models

Decision making is undeniably an essential and vital part of the human life. Very often, numerous smaller decisions have to be made before a decision can eventually arrive at its best conclusion. Moreover, each of these decisions may directly or indirectly have a bearing on other subsequent decisions and can easily influence the overall decision and conclusion. However, most decision making processes and systems treat each of these decisions as independent and unrelated decisions and obscure the decision makers from seeing the true effects and influence of each decision. It is also unreasonable and impractical to expect decision makers to operate a different decision making system for each individual and smaller decision and comprehend the full effects of the consolidation and integration from these individual decisions. Furthermore, the decision makers are often restricted by the dictatorship of the decision making systems that lay down the steps and techniques behind analysing and solving the problem. This defeats the purpose of a decision system that is meant to assist a decision maker in any decision making processes. Regardless of the complexity nature or simplicity prospective of certain decision makers, a decision together with its interrelated decisions must be represented in a simplistic manner for decision makers to read, understand and communicate. This calls for the use of models to represent, describe and depict the problem and its interaction under consideration (Eppen and Gould, 1984; Golub, 1997).

Models are abstractions of the problem under consideration. Models can also be actual model instances of the abstraction schema (Krishnan and Chari, 2000). They can also be executable computer program modules that are used to generate model solutions (Krishnan and Chari, 2000) while with the intention of providing insights rather than numbers (Geoffrion, 1989). It is important to note that a model is said to be useful only if it is represented in some medium, hence, much description must be captured so that it can be decomposed into sufficient detail for model execution (Curtis, Kellner and Over, 1992). Modelling is the process of understanding, capturing, representing, and solving such models (Brewer and DeLeon, 1995; Eriksson, 2003). Modelling is an important process since it is intended to capture functional, behavioural, organisational, and informational perspectives (Curtis, Kellner et al., 1992). However, most systems focus only on the functional and informational perspectives at the detriment of the organisational and behavioural perspectives. Furthermore, most decision systems are observed to concentrate on efficiency, effectiveness, interfaces or data depending on the disciplines of the system designers.

The inclusion of simply a traditional optimising (Geoffrion, 1987; Draman, Kuban Altinel, Bajgoric, Tamer Unal and Birgoren, 2002) or a satisficing paradigm method but not both prevents the possibility of integrating optimising and satisficing in the process. Furthermore, most decision systems are observed to solve problems by receiving decision parameters at the start of the process. The lack of intervention during the process suggests that only fixed sequential decisions are considered. Despite the fact that many decisions execute in a sequential fashion (Simon, 1983) the order of execution should be neither fixed nor predetermined. This way, the decision systems are flexible and able to solve complex problems that consist of interrelated decisions in various domains and/or paradigms. Moreover, the problem can be recognised and modelled differently by different user group, but the user must be able to progress from novice to competent users (Dreyfus and Dreyfus, 1986). Such decision systems need not be complex. On the contrary, they must be simple enough for a novice user to operate, and flexible enough for a competent user to incorporate and integrate multiple decisions.

To overcome these problems mentioned, we first propose a converging decision analysis process, individual as well as organisational decision learning models, and a cyclical modelling process. We then propose a Flexible Object-Oriented Decision System (FOODS) framework and architecture to support these models and processes. A prototypical system was developed and implemented to act as a proof-of-concept to support the proposed processes, framework and architecture. Object-oriented concepts were leveraged as they are better at handling complex problems that consists of interrelated components. One-Dimensional Cutting Stock Problem (1D-CSP) was used as an example though 1D-CSP was considered to be a simple problem in pure mathematical terms but it becomes a reasonably complex once one considers all the real world constraints and interrelated decisions involved in the process.

Individual and Organisational Decision Making Processes

A decision process that can flexibly solve problems in any domain or paradigm is essential in decision making and support. Often a decision comes with many factors. Therefore, decision making can be improved by first focusing on essential factors, and eliminating non-essential ones. This is known as the attention-focused method that provides a cut down version of the problem rather than an actionable result (Holtzman, 1989). Subsequently, a decision can be derived from the reduced problem so as to provide an actionable result. This is the decision-focused method that produces an actionable result from the given problem (Holtzman, 1989).

Often in solving a complex problem, many intermediate decisions are necessary for evolution and learning to take place before recommending on a course of action. If the result or outcome of an intermediate decision is not satisfactory it can be revisited. This is the iterative process known as the convergence process (Langley, Mintzberg, Pitcher and Posada, 1995).

We propose a new converging decision analysis process to allow for many intermediate decisions to be made through the two differing focused methods, starting from "concentrating" on essential factors and then "deciding" on given factors. Each decision converges as the scope of the problem is narrowed through the process of evolution and learning from decisions over time and refined through several iterations of attention-focused and decision-focused methods within the decision making process before prescribing or settling on a course of action. Such a convergent view is important not only because there are interrelated decisions within a complex problem but that these decisions may evolve, narrow and refine over time as decision makers become more certain about them. This proposal is depicted in Figure 6 and illustrated using an 1D-CSP example. The patterns generation heuristic concentrates on generating the desired combination of cutting patterns, while the decision-focused method is subsequently employed to determine the selection of patterns among the generated patterns. The decision problem then converges during the creation of the linear programming constraints that identifies the feasible area of the problem under consideration. A decision-focused method is subsequently employed to find the optimal point within the feasible area.

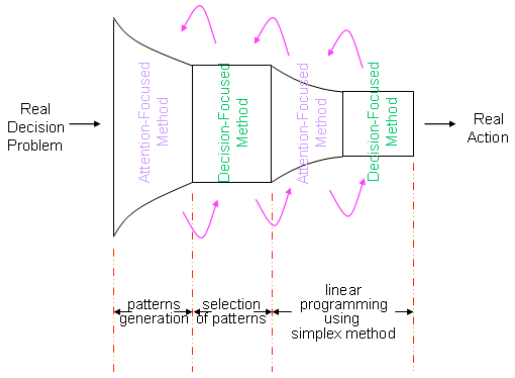


Figure 6: Converging Decision Analysis, as in an 1D-CSP scenario

Such process also allows interrelated decisions to be assessed and revisited through the various iterations. Furthermore, the course of action derived can be an optimised or a satisficing solution. An attention-focused method or a decision-focused method does not necessarily produce an optimal or a satisficing solution. It is up to the decision maker to decide on what sort of solution is desired at the time and as the decision maker learns more about the decisions and problems on hand. Each decision and solution is encompassed in a decision model. Each actionable scenario may be fed into another decision model to produce another optimal or satisficing action. In a complex problem that consist of interrelated decisions this process may repeat over several scenario instances before an ultimate action is reached, where each of them can take on a different solution option. Each decision model may return to itself for refinement, or return to the previous model for additional processing, or feed to the next model for further processing. This return may be due to infeasible solution, or a better understanding of the problem/model which lead to a change in the parameters of the model.

The fact that each decision and solution is encompassed within a decision model that considers both optimising and satisficing solution methods suggest that a learning model can

exist. This means that any information being fed into the decision model, it can repeatedly go through the optimising and/or satisficing solution methods a number of times until an actionable scenario is derived and produced. For example, in an 1D-CSP example, a satisficing method is first adopted to immediately narrow down the cutting patterns, followed by a satisficing heuristic to select the generated cutting patterns to be considered. From there, an optimising method can be applied to decide on precisely which cutting patterns are used and for how many occurrences. This example scenario is also captured within the decision model in Figure 7. This decision model in terms of its individual and organisational learning can be expanded to be applicable in an opened environment where a single-loop as well as a double-loop learning takes place (Argyris and Schön, 1996), as depicted in Figure 7. The theories of action in use are essentially a decision problem. Besides the imperfect information supplied at the beginning in Phase 1, additional information as in Phase 2 may be requested pending on whether additional factors are applicable in a particular decision problem. Such expansion and shows that an actionable scenario resulted from the decision model may be identified as correctable errors as in Phase 4 that may become a new source of information, or as uncorrectable errors as in Phase 4 that are evaluated against fundamental objectives, espoused models, and reality as in Phase 5 that in turn correct the model in use as in Phase 3. This process is said to be in an opened environment because the underlying model can be altered as a result of the proposed action. Such decision models are also consistent with the converging decision analysis proposed earlier and as shown in Figure 6.

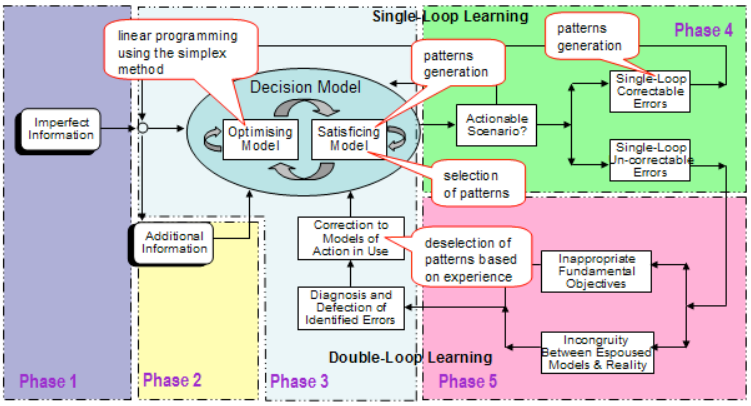


Figure 7: Individual and Organisational Decision Learning, as in an 1D-CSP scenario

Cyclical Modelling Process

To support the decision making process proposed we need to have a modelling process that integrates ideas and themes from many processes proposed in literature (Eppen and Gould, 1984; Sage, 1991; Mathur and Solow, 1994; Brewer and DeLeon, 1995; Argyris and Schön, 1996; Golub, 1997; Krishnan and Chari, 2000). A new modelling process that is cyclical and iterative, and enables continuous adjustment and refinement is proposed, and is summarised in Figure 3. It is also especially valuable on modelling the system components of the decision. Once a problem is understood it can be represented in the form of a model which is then instantiated with data and integrated with solvers so that it can be executed. Such a model is especially beneficial if it is storable and retrievable for later use and comparison. Once a model is represented, a solution can be derived through analysing and investigating it as well as comparing with various model instances. The derived solution is then reviewed and

validated. If it is considered unsatisfactory such information can be used to modify and reformulate the decision model. This is the proposed cyclical modelling process.

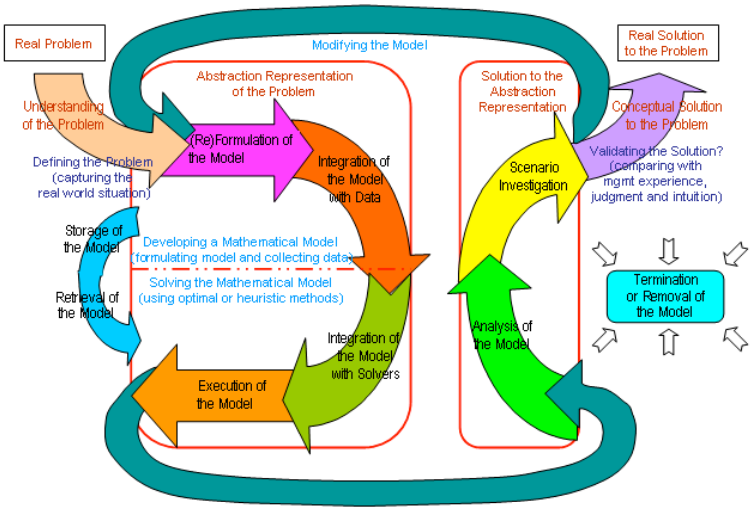
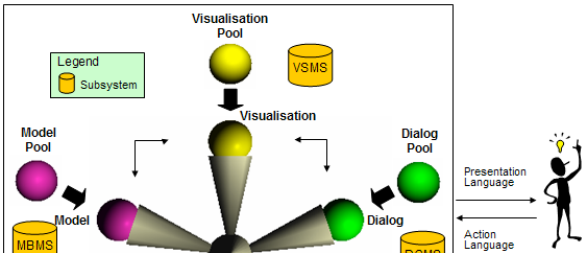


Figure 8: Cyclical Modelling Process

In reality, problems are often considered to be complex because the result from one decision can influence a subsequent decision. It may at times have an indirect bearing on subsequent decisions. Therefore it is important that all such decisions be captured but be structured in such a flexible manner that enables the decisions to be considered and solved differently each time. Each of these decisions can be represented in the form of a model, whether it is simply an abstraction schema, actual model instance, or executable computer program module. It may also be treated as a permanent and independent scenario model which can be retrieved and included as part of a bigger scenario problem. Alternatively, it may be treated as a temporary scenario that is pipelined within a bigger scenario problem. Such model integration treatments are subject to the discretion of decision makers at the time of making such decisions. Thus, a problem is considered complex since it may encapsulate many levels and depths of models integration, such as pipelining, recursion, consolidation, and splicing. However, models that are pure abstractions are insufficient (Curtis, Kellner et al., 1992); they must be populated with raw data that instantiates the model, solved with some known methods/solvers through an user interface that enables decision makers to dialog with the system; and presented using visualisation.

Decision Support and Modelling Framework and Architecture

We propose a flexible object-oriented decision system (FOODS) framework to support the decision making and modelling processes and overcome the problems and issues mentioned earlier. The FOODS framework comprises of five major components, namely, data, model, dialog (Sprague, 1980), solver (Krishnan and Chari, 2000), and visualisation (Chen, 2000). Each of the components is briefly described in this section.



Data is the component where the raw data is physically stored and retrieved from. Model is the scenario instance that represents the problem under consideration. Solver is a component that comprises a combination of known mathematical and computational methods used to solve a model instance. Visualisation is a component that combines both embedded text and chart presentations. Dialog is an interface template that bridges language exchange and user level interaction, which is defined somewhat differently from what the component is first introduced by (Sprague, 1980). Each of the components is independent from one another, but they are able to map to each other (Ramirez, Ching and St Louis, 1990) as a pluggable object (Wrobel, Wettschereck, Sommer and Emde, 1997). Each of the components is also retrievable from their component pool (Chen, 2000) and can be integrated through the kernel for scenario execution. In order to ensure that each component is independent from each other, generic modelling ideas and integration issues are explored and considered, such as, data-model, model-solver, solver-visualisation, data-visualisation, and data-solver independences (Ramirez, Ching et al., 1990).

Figure 9: Flexible Object-Oriented Decision System Framework

Each decision scenario calls for a different combination of components (Fierbinteanu, 1999). Since the result from one decision may influence a subsequent decision, it is therefore important to store certain decisions as independent and permanent model instances so that they can be pipelined or consolidated as part of another decision model. The interrelated nature of decisions makes a problem less straightforward and highlights the representation and integrations of models that bring reduced cost (Dolk and Konsynski, 1985), increased modelling productivity and reusability (Geoffrion, 1989), facilitated growth and evolution of modelling systems, and improved managerial decisions (Tung, Ramirez and St Louis, 1991; Bhatt and Zaveri, 2002). Depending on the nature or condition of the problem, the existence of the integration may be temporary or permanent. A model builder may decide on the suitability of the type of models at the time of execution. A combination of temporary and permanent integration may also be desired if the problem consists of numerous smaller decision models. For example in Figure 10, a model may consist of a data D1 and a solver component S1, as illustrated by model M1, and can be saved as a retrievable scenario for later use, and hence is permanent in its integration; whilst another model may consist of two data components D2 and D3, a solver S2, a visualisation V, and a dialog component G, as illustrated by model M2. Furthermore, the result from a model component may be an input to another model component, as illustrated by M1 feeding into M2. Hence, the combination of data D2, D3, dialog G, visualisation V, solver S2, models M1 and M2 can be seen integrating temporarily. This example instance shows that the decision making and modelling processes can be supported flexibly through model integration.

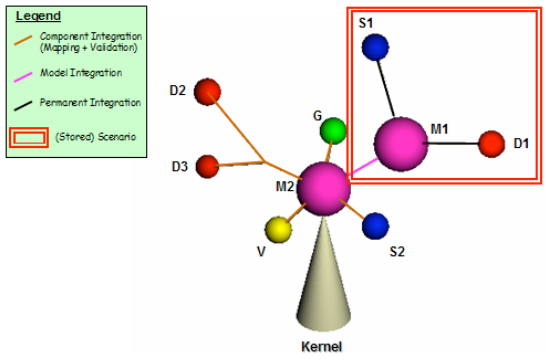


Figure 10: An Example Instance

The proposed framework focuses on flexible modelling with loose coupling as well as through model integration. All four levels of models integration (Kottemann and Dolk, 1992), namely, the consolidation of values obtained from different models, the consolidation or pipelining of models that belong to the same modelling paradigm, the pipelining of models that belong to different modelling paradigm and the splicing of models are considered in the framework. A model builder may decide to store the output of that model before feeding it into the next model, or allow reformulation and recalculation to happen at each occurrence. This level of integration is often not developed because it is difficult to put into practice and is seldom supported by any implementation environment.

The search for a new proposed architecture is necessary since the integration among components and models is an important issue among more challenging problems. The new proposed architecture although rooted in tradition (Sprague, 1980; Chen, 2000), expands and enhances existing work, and are organised into four distinct layers, namely, presentation layer, kernel integration layer, component integration layer, and component pool layer, as pictured in Figure 11.

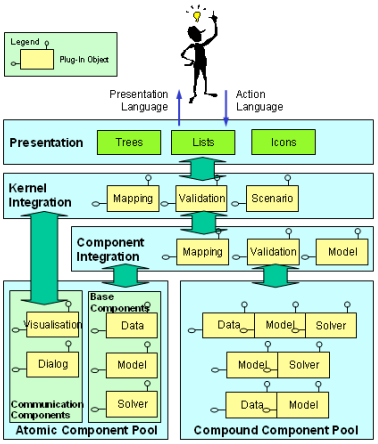


Figure 11: Flexible Object-Oriented Decision System Architecture

The presentation layer is the layer that communicates with the users in the form of a presentation language, and receives instructions from the users in the form of an action language (Chen, 2000). The kernel integration layer is the layer that takes care of the kernel, the composition scenario of the kernel and the execution of the kernel. It holds all the components in place and executes in the form of a problem scenario, and hence is able to handle stored scenarios being included as an independent scenario inside a bigger scenario. The component integration layer is the layer that ensures the amalgamation of various components. This layer focuses on component integration as well as model integration that occurs on an ad hoc basis until it is stored. The component pool layer manages the components which hold data and functionalities (Chen, 2000). There are two types of components in this layer: compound components, and atomic components. Compound components are component objects that are permanently integrated, and can be treated as stored scenario objects that are pluggable and be included as part of another bigger scenario. Atomic components can be categorised as base components that exist in most scenarios and can be permanently integrated into a compound component; and communication components that are essential if the description or results of the problem scenario is intended to be communicated. Each component or sub-component is constructed and implemented using object-oriented concepts, is signified by an object-oriented class, and contains its own properties and functionalities.

Through the use of the FOODS architecture, a decision maker can undertake the decision making process one phase at a time by creating smaller scenarios and integrating existing

scenarios into a bigger scenario. This allows the decision maker to have a better appreciation of the effects of a decision within the bigger context of the overall decision.

Decisions and Modelling Systems

A prototypical system was implemented to prove the validity of the proposed processes, FOODS framework and architecture by focusing on flexible modelling with loose coupling. This meant that the values for any coefficient within a constraint, or a parameter can be changed, and the number of problem constraints, parameters, and objectives can be adjusted accordingly. In addition, the result from one model can be used as an input to another model, the problem is not restricted to one computational direction, and different scenarios can be investigated and compared. The essence of flexible modelling in this research and prototype is to allow decision makers to decide in a non-fixed and predefined manner that enables refinement and intervention, as well as to learn and improve on existing instances or future scenarios. The prototypical system was implemented using an object-oriented database management system following the object-oriented concepts: abstraction, encapsulation, inheritance, and reusability, and was in-line with (Fourer, 1997) mathematical structure that addressed sparsity. In order to illustrate the simplicity and complexity of interrelated decisions within the implemented prototypical system, 1D-CSP was used as an example. Four differing example scenarios that distinguish each other in the context of versatility and autonomy were used to assess and validate. For the purpose of illustrations, only the automated and manual scenarios are presented in this paper.

Each of the example scenarios is completed by stepping through the steps of the proposed cyclical modelling process as each activity, decision, and request occurs in a specific phase of the decision-making and modelling process as denoted in the cyclical modelling process. There are several interested parties that make up the stakeholders in the 1D-CSP and should not be determined by the roles they play, but are rather grouped accordingly to their competencies and user permissions: decision maker, programmer, and model builder (Sprague, 1980). Despite the fact that the roles they play have specific tasks, the parties involved must be able to progress from novice users to competent users (Dreyfus and Dreyfus, 1986). These stakeholders in turn interact with the activities, decisions, and requests as represented in the form of use cases. These use cases are clustered accordingly to the decision making process and decision model as depicted in Figure 7. These five use case clusters, namely, domain, optimising, satisficing, utilities, and generic, are shown as column labels in Figure 12 and Figure 14. Hence, Figure 12 and Figure 14 also show the grouping of use cases according to the decision-making and the modelling process, as well as the use case clusters and the user groups. The remainder of this section illustrates the 1D-CSP decision-making process of two different scenarios. The first one is an automated scenario. The second scenario is a manual scenario.

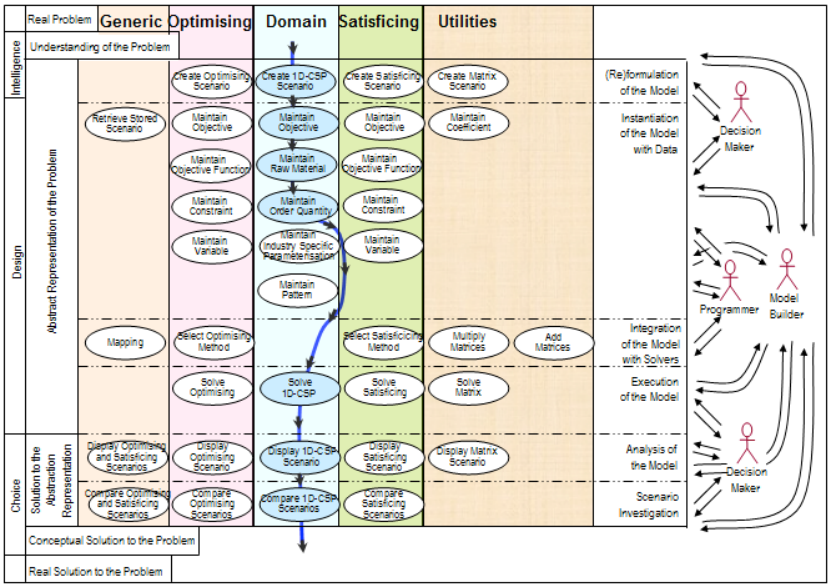


Figure 12: The Modelling Process of an Automated 1D-CSP Scenario

Automated Scenario

An automated scenario is defined as a model scenario in which the decision maker enters the parameters required in the domain and instructs the system to solve and display the problem. The decision maker does not necessarily know the execution order of the decision-making process and does not interfere with it. S/he only interacts with the system about the description of the problem and the presentation of the solution. Furthermore, the decision paths of this scenario usually lie along the domain route and this scenario should only be available as a typical choice for a novice or an advanced beginner. Figure 12 gives a quick guideline as to the decision paths and the modelling process in an automated 1D-CSP scenario.

Formulation of the Model and Instantiation of the Model with Data

The user starts the modelling process by creating a 1D-CSP model instance, and selecting an objective. The user is also able to select more than one objective. Next, the raw material widths that are available for production are entered. Then, the demand orders are keyed in with the maximum and minimum quantities required. The orders need not be in a decrementing order. There is also no limit on the number of permissible demand orders.

Execution and Analysis of the Model

Once the required data of the domain is entered, the user can solve the 1D-CSP and display the solution. The user can also change the value of any entered data for recomputing another scenario instance. The solution is shown in the "run-times" column in Figure 13. The user will see only the recommended cutting patterns.

Patterns / Constraints:

#	45m	36m	31m	14m	edge	chosen?	run-times
1	1	1		1	5	YES	97
2		1	2		2	YES	198
3		2		2	0	YES	158

Figure 13: Display 1D-CSP Scenario

Manual Scenario

A manual scenario is defined as a model scenario which the model builder has the total freedom to design the steps and sequence in which the decision making processes should be actioned and executed. The model builder also has the freedom to choose any solution methods and available solvers. Such a manual scenario may also consist of several stored scenarios which are temporarily integrated together. The fact that the model builder has the freedom to capture and design the problem and choose the solution methods suggests that a manual scenario is domain and paradigm independent. Hence, the model builder is able to create an 1D-CSP scenario, a pure optimising scenario such as a linear programming scenario, or a satisficing scenario using the same prototypical system. This scenario should only be available as a typical choice for a model builder or an advanced user. Figure 14 gives a quick guideline as to the decision paths and the modelling process in a manual 1D-CSP scenario.

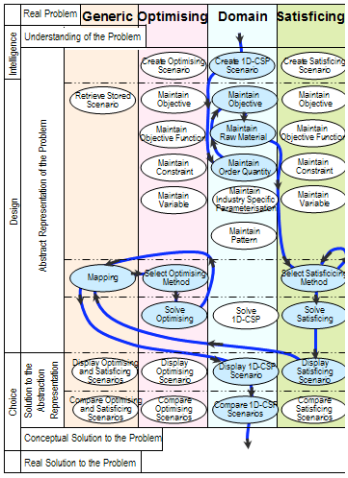


Figure 14: The Modelling Process of a Manual 1D-CSP Scenario

Formulation of the Model

The model builder starts the modelling process by creating a domain model instance (as shown in Figure 15). This domain scenario happens to be of the same type as the satisficing scenario and therefore minimises mapping between domain and satisficing scenarios.

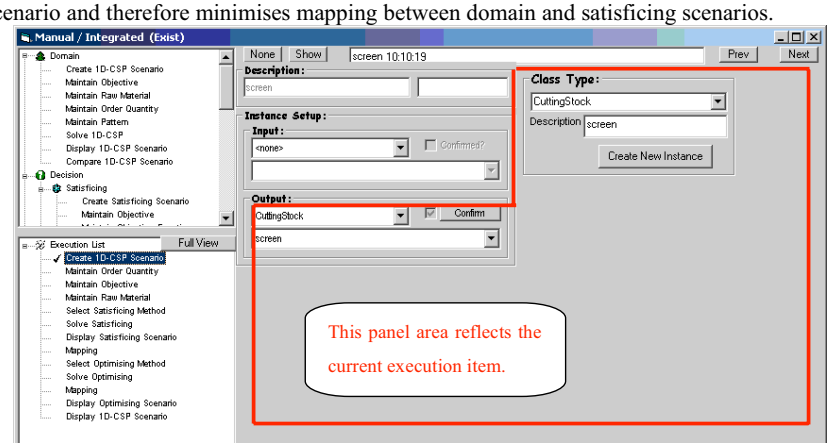
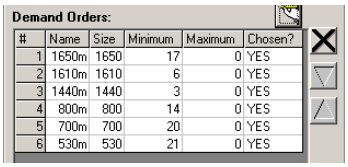


Figure 15: Create Model Instance

Instantiation of the Model with Data

The model builder continues on with the modelling process and is able to decide on the order of entering the data as required by the domain that best suit him/herself, or to recall and modify a previously stored scenario data. In this instance, the orders and quantities are first entered (as shown in Figure 16), followed by the objective, and the raw material availability.



#	Name	Size	Minimum	Maximum	Chosen?
1	1650m	1650	17	0	YES
2	1610m	1610	6	0	YES
3	1440m	1440	3	0	YES
4	800m	800	14	0	YES
5	700m	700	20	0	YES
6	530m	530	21	0	YES

Figure 16: Maintain Order Quantity

Integration of the Model with Satisficing Solvers

Once the domain parameters are recorded, the model builder selects a satisficing method for execution based on his/her personal knowledge on generating some suitable cutting patterns to start the decision making and modelling process.

Execution of the Model

Once a solver is chosen the model builder can execute and solve the instantiated satisficing model instance. The model builder may wish to return to the previous step and repeat the process. This is done so that s/he can correct the satisficing suggested model based on his/her own knowledge of the practical limitations.

Analysis of the Model

The model builder is then able to view the solved solution and come to a conclusion on whether to adjust the inputs or outputs, or to continue with the recommended solution. The solved cutting patterns solution can be viewed in a matrix style or in a stacked bar chart to visually evaluate the output quantities and wastage (as shown in Figure 17).

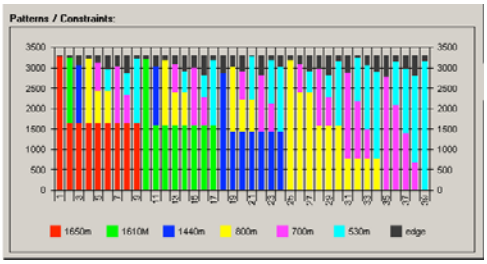


Figure 17: Display Satisficing Solution

Integration of the Model with Optimising Solvers

In order to resolve the model instance in an optimised manner, a new optimising scenario is created and the current model instance is mapped to the new optimising model as they are of different data type structure. This integration is in line with the optimising-satisficing decision model depicted in Figure 7. Once it is mapped, the model builder selects an optimising method for execution. This optimising method first concentrates on generating the relevant demand constraints given the already predetermined cutting patterns, and then decides on the best frequencies of each cutting patterns given the specified objective(s). The narrowing and deciding actions described here support the attention-focused and decision-focused methods.

Execution of the Optimising Model

Once a solver is chosen the model builder can execute and solve the instantiated optimising model instance.

Integration of the Optimising Model with the Domain Solvers

In order to interpret the optimised model solution, the instantiated and solved optimising model instance is mapped back to the original domain model instance for viewing or further processing.

Analysis of the Domain Model

The model builder is now able to view the results from both satisficing and optimising solution methods and come to a decision on whether the inputs or outputs need adjustment, or simply to continue with the recommended solution. The frequencies of each prescribed cutting patterns solution can be viewed in a matrix style (as shown in Figure 18) or in a pie chart to visually evaluate the recommended cutting patterns and the overage usage of these patterns.

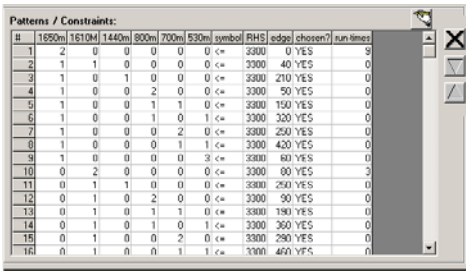


Figure 18: Display Solution of Model Instance

Scenario Investigation

Once a solution is derived, it can be compared to other similar stored scenarios for the best practical implementation.

Comparing Automated and Manual Scenarios

It is important to note several differences between the two scenarios based on the previous sample sessions and discussions. The first one being that the user does not decide or integrate with any solvers in the automated scenario. The second one being that the user does not decide on the execution order or revisit any steps of the decision making process in the automated scenario. The third one being that the user has to decide and devise every single detail including the mapping data types in the manual scenario. The fourth one being that the user can retrieve data already stored or computed in the manual scenario.

It was discussed at the start of this section that in order to support the proposed processes, framework, and architecture, the prototypical system need to focus on the flexible modelling with loose coupling. We can clearly see through the illustrations that the manual scenario exhibits flexible modelling principles and supports interrelated real world scenarios better than the automated scenario. For example, the manual scenario can flexibly increase or decrease the number of parameters involved in the scenario problem; change the computational direction of the scenario problem; group the result from models together or use the result from one model as an input to another model; affect the choice of subsequent models based on the result of previous model; and eases the investigation of different scenarios. Some of these principles are the same as those presented in the four levels of models integration that are discussed earlier. A checklist of these high-level research objectives is summarised in Figure 19 and is addressed in the two scenario examples.

	Description of Detailed Research Objectives	Principles of Flexible Modelling	Automated Scenario	Manual Scenario
Simple ↑ COMPLEXITY	❖ the values can be changed	value (data) independence	section 5.1	section 5.2
	❖ the number of problem constraints (i.e. rows) can be in/decreased	dimension (data) independence	section 5.1	section 5.2
	❖ the number of parameters (i.e. variables or columns) can be in/decreased	dimension (data) independence		section 5.2
	❖ the number of objectives can be in/decreased, which in turn affects the parameters	dimension (data) independence	section 5.1	section 5.2

Figure 19: Summary of Flexible Modelling Between the Two Scenarios

Conclusion

The basic concern of any decision system is to ensure that decision makers are supported to make better decisions, rather than being replaced and/or dictated by decision systems. However, many real life problems under consideration are often interrelated and are in need of capturing the interaction and influence between each interrelated decision. A simple mathematical example such as 1D-CSP was used to illustrate the interrelated nature of decisions through the proposed decision making and modelling processes, framework, and architecture. This is done by investigating the flexibility and complicity of decision making processes and which consist of several interrelated decisions.

The implemented prototype demonstrates some art of decision making. Despite the predictable phases in decision making, the actual steps of decision making are frequently carried out not in a strict order. The proposed converging decision analysis expresses the somewhat unpredictable steps in decision analysis and decision making, and articulates the needs for attention-focused and decision-focused decision in each subsequent step. Furthermore, individual and organisation learning may occur during a decision making process and warrants the need to revisit and re-adjust the model accordingly or completely modify and construct a new model. Moreover, the decision maker should be free to choose either an optimising or a satisficing method to resolve a problem, rather than be restricted to one method or the other. The evaluation results and sample sessions show that the proposed decision making and modelling processes, framework, and architecture are generic and are effective in assisting decision makers in solving their everyday problems that consists of interrelated decisions and under varying conditions of versatility and autonomy.

References

Argyris, C. and D. Schön (1996). *Organizational Learning II: Theory, Method and Practice*. Mass, Addison Wesley.

Bhatt, G. D. and J. Zaveri (2002). The Enabling Role of Decision Support Systems in Organizational Learning. *Decision Support Systems*. Vol. 32, No. 3: 297-309.

Brewer, G. D. and P. DeLeon (1995). *The Foundations of Policy Analysis*. Homewood, IL, Dorsey Press.

Chen, K. J., Sundaram, D., and Srinivasan, A. (2000). The Design and Implementation of a Decision Support System Generator. *IFIP8.3 Conference on Decision Support Through Knowledge Management: Position Papers on Future Directions in Decision Support Research*, Stockholm, Sweden.

Curtis, B., M. Kellner and J. Over (1992). Process Modelling. *Communications of the ACM*. Vol. 35, No. 9: 75-90.

Dolk, D. and B. Konsynski (1985). Model Management in Organizations. *Information & Management*. Vol. 9: 35-47.

Draman, M., I. Kuban Altinel, N. Bajgoric, A. Tamer Unal and B. Birgoren (2002). A Clone-Based Graphical Modeler and Mathematical Model Generator For Optimal Production Planning in Process Industries. *European Journal of Operational Research*. Vol. 137, No. 3: 483-496.

Dreyfus, H. L. and S. E. Dreyfus (1986). *Mind Over Machine: The Power of Human Intuition and Expertise in the Era of the Computer*. New York, Free Press.

Eppen, G. and F. Gould (1984). *Introductory Management Science*. New Jersey, Prentice-Hall.

Eriksson, D. M. (2003). A Framework for the Constitution of Modelling Processes: A Proposition. *European Journal of Operational Research*. Vol. 145, No. 1: 202-215.

Fierbinteanu, C. (1999). A Decision Support Systems Generator For Transportation Demand Forecasting Implemented by Constraint Logic Programming. *Decision Support Systems*. Vol. 26: 179-194.

Fourer, R. (1997). Database Structures for Mathematical Programming Models. *Decision Support Systems*. Vol. 20: 317-344.

Geoffrion, A. (1987). An Introduction to Structured Modelling. *Management Science*. Vol. 33, No. 5: 547-588.

Geoffrion, A. (1989). Reusing Structured Models via Model Integration. *Twenty-Second Annual Hawaii International Conference on System Sciences*, Los Alamitos, CA, IEEE Computer Society Press. 601-611.

Golub, A. L. (1997). *Decision Analysis: An Integrated Approach.* New York, John Wiley & Sons.

Holtzman, S. (1989). *Intelligent Decision Systems*, Addison-Wesley Publishing.

Kottemann, J. E. and D. R. Dolk (1992). Model Integration and Modelling Languages: A Process Perspective. *Information Systems Research.* Vol. 3, No. 1: 1-16.

Krishnan, R. and K. Chari (2000). Model Management: Survey, Future Research Directions and a Bibliography. *Interactive Transactions of ORMS*. Vol. 3, No. 1.

Langley, A., H. Mintzberg, P. Pitcher and E. Posada (1995). Opening up Decision Making: the View from the Black Stool. *Organization Science*. Vol. 6, No. 3, May-Jun 1995: 260-279.

Mathur, K. and D. Solow (1994). *Management Science: The Art of Decision Making*. Englewood Cliffs, New Jersey, Prentice Hall.

Ramirez, R. G., C. Ching and R. D. St Louis (1990). Model-Data and Model-Solver Mappings: A Basis for an Extended DSS Framework. *ISDSS Conference Proceedings.*

Sage, A. P. (1991). Chapter 6: Information Processing in Individuals and Organizations. Decision Support Systems Engineering, John Wiley & Sons**:** 205-262.

Simon, H. (1983). *Models of Bounded Rationality*. Cambridge, M.A., Harper and Row.

Sprague, R. J. (1980). A Framework for the Development of Decision Support Systems. *MIS Quarterly*: 1-26.

Tung, L., R. G. Ramirez and R. D. St Louis (1991). Model Integration In An Object-Oriented Model Management System. *IEEE*. Vol. 73.

Wrobel, S., D. Wettschereck, E. Sommer and W. Emde (1997). Extensibility in Data Mining System. *2nd International Conference on Knowledge Discovery and Data Mining*. 214-219.